Exp – 2

REGNO-RA1911026010119

NAME-SANKALP JAIN

Graph Coloring problem

Lab-2

RA1911026010119 Sankalp Jain

Aim → To Solve & perform Vertex & edge coloring

Algorithm –

① Vertex coloring

→ Add list of colors.
→ Then give list of graph edges & nodes and create function to built graph from edges.
→ convert edges to undirected graph.
→ Create function to assign colors to vertex.
→ Check for 1st color and assign all others
→ Loop West Vertex and color it with lowest numbered color That Was been colored on any Vertices Adjacent to it.
→ Print.

② Graph edge coloring

→ Defin graph by Number of Vertex & edge connected by each other. then create function to determine edge colors.
→ Traverse all edges one by one & check them.
→ If color Vertex found to be negative then assign color and increment color i.e change color for West.
→ Add it to the set
→ Repeat until all edges of graph are colored

Code:

VERTEX COLORING

```
class Graph:
    def __init__(self, edges, n):
        self.adjList = [[] for _ in range(n)]

        # add edges to the undirected graph
        for (src, dest) in edges:
            self.adjList[src].append(dest)
            self.adjList[dest].append(src)



def colorGraph(graph, n):
    result = {}
    for u in range(n):
```

```
        assigned = set([result.get(i) for i in graph.adjList[u] if i in
result])
        color = 1
        for c in assigned:
            if color != c:
                break
            color = color + 1



        result[u] = color


    for v in range(n):
        print(f'Color assigned to vertex {v} is {colors[result[v]]}')



if __name__ == '__main__':


        colors = ['', 'GREEN', 'ORANGE', 'RED', 'CYAN', 'PURPLE',
'PINK',
            'PINK', 'BLACK', 'GREEN', 'CYAN', 'BLUE']
    edges = [(0, 1), (0, 2), (1, 5), (2, 5), (2, 4), (2, 3), (0, 3), (1, 4)]
```

```
n = 6

# build a graph from the given edges
graph = Graph(edges, n)
colorGraph(graph, n)
```

EDGE COLORING

```
from queue import Queue
def colorEdges(ptr, gra, edgeColors, isVisited):
    q=Queue()
    c = 0

    colored=set()

    if (isVisited[ptr]):
        return

    # Mark the current node visited
    isVisited[ptr] = True
```

```
for i in range(len(gra[ptr])) :

        if (edgeColors[gra[ptr][i][1]] != -1):
                colored.add(edgeColors[gra[ptr][i][1]])



for i in range(len(gra[ptr])) :

        if not isVisited[gra[ptr][i][0]]:
                q.put(gra[ptr][i][0])

        if (edgeColors[gra[ptr][i][1]] == -1) :

                while c in colored:

                        # increment the color
                        c+=1
```

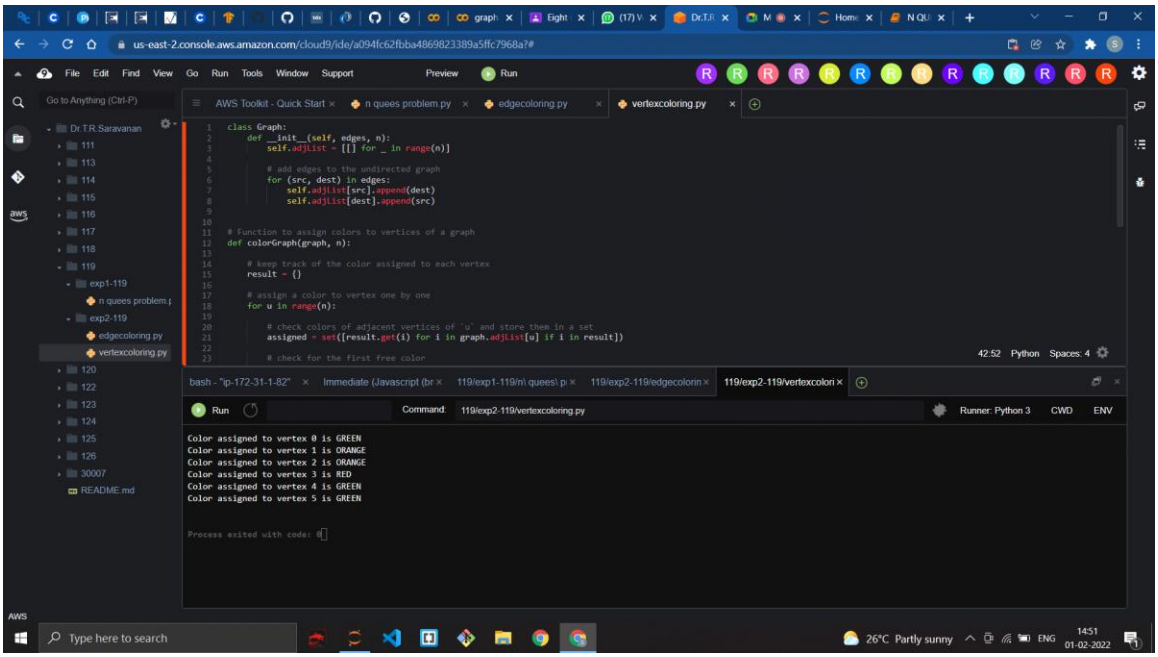
```
                # copy it in the vector
                edgeColors[gra[ptr][i][1]] = c

                # then add it to the set
                colored.add(c)
                c+=1




    while not q.empty() :
        temp = q.get()

        colorEdges(temp, gra, edgeColors, isVisited)


    return


# Driver Function
```

```
if __name__=='__main__':
    empty=set()

        gra=[]

    edgeColors=[]

    isVisited=[False]*100000

    # Enter the Number of Vertices
    # and the number of edges
    ver = 4
    edge = 4

    gra=[[] for _ in range(ver)]
    edgeColors=[-1]*edge

    gra[0].append((1, 0))
    gra[1].append((0, 0))
```

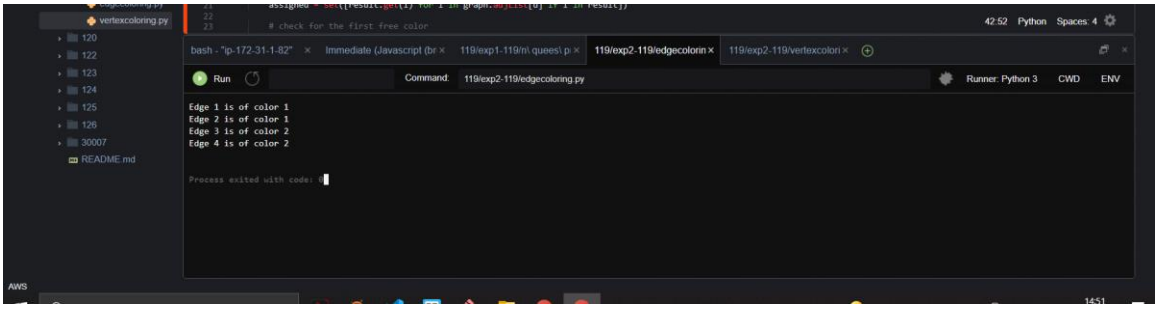
```
    gra[1].append((2, 0))
    gra[2].append((1, 1))

    gra[2].append((1, 2))
    gra[3].append((2, 2))

    gra[0].append((0, 3))
    gra[3].append((0, 3))

    colorEdges(0, gra, edgeColors, isVisited)

    # printing all the edge colors
    for i in range(edge):
        print("Edge {} is of color {}".format(i + 1,edgeColors[i]
+ 1))
```

Output:

Vertex coloring:



Edge coloring:



Result:

Hence Graph Coloring problem for both vertex coloring and Edge coloring were studied and solved.